ARDUINO LIGHTNING DETECTOR AND DATA MONITOR

Nick Cinquino, KC9MKW, May 2015

The Radio Frequency emission of natural lightning is very broad, spanning the ELF below 1KHz to over 100MHz (1). Recent research shows that a rare lightning event may be preceded by Gamma ray bursts, called Dark Lightning, extending the EM spectrum even further. Peak RF emissions are in the 10 KHz range, but there is still plenty of energy available in the 500KHz range (AM Broadcast Band) for DIY lightning data analysis.

For this lightning detection method, a 1920's style crystal radio circuit, with excellent selectivity, is combined with the data handling capability and speed of the Arduino microcontroller!

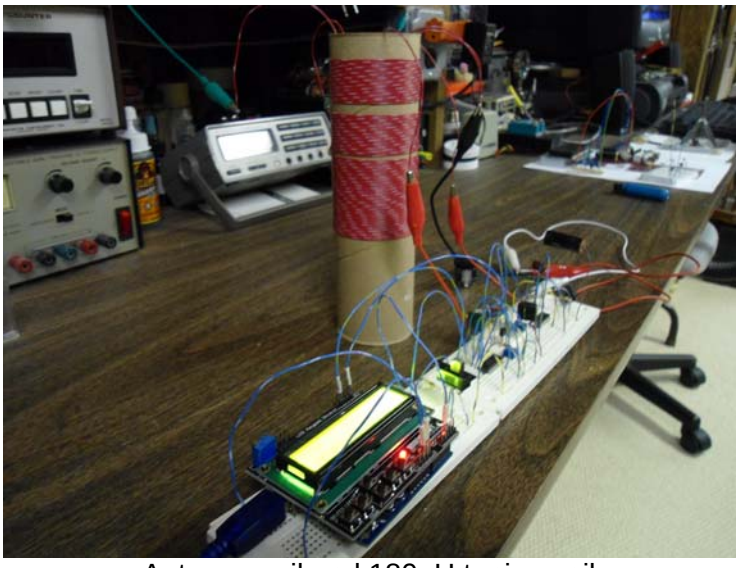
The heart of the radio receiver circuit is the coil; a coupled antenna coil and tuning coil with tap. This is made on a 5.5cm (2.16") paper tube; a piece of similar size PVC pipe will be fine. The diameter does not have to be exact, nor does the number of turns…one or 2 missing or extra turns will still work.
The antenna coil is 25 turns of 24 gauge insulated wire, of 1mm diameter with insulation, wound with no gaps between turns (see pic). One end of this antenna coil is attached to an "aerial" of 2' or more (keep it indoors, *especially* during thunderstorms!), and the other end attaches to a good earth ground like a water pipe. Just below the antenna coil, wind another 53 turns, same wire type. Add a tap, and continue with another 25 turns. The actual tuning coil is the entire 78 turns. The theoretical inductance of this coil is 180uH. If you want to design a coil for a different diameter tube, use the Wheeler 1925 long coil formula:

$$L = a^2 N^2 / (9 a + 10 b)$$

Where L = microhenries, a = radius inches, N = number of turns, and b = length, inches.
Then use the tank resonance calculation for various settings of the tuning cap, 20-360pF:

$$f_{resonant} = \frac{1}{2\pi \sqrt{LC}}$$

Where f = Hertz, L = Henries and C = Farads.

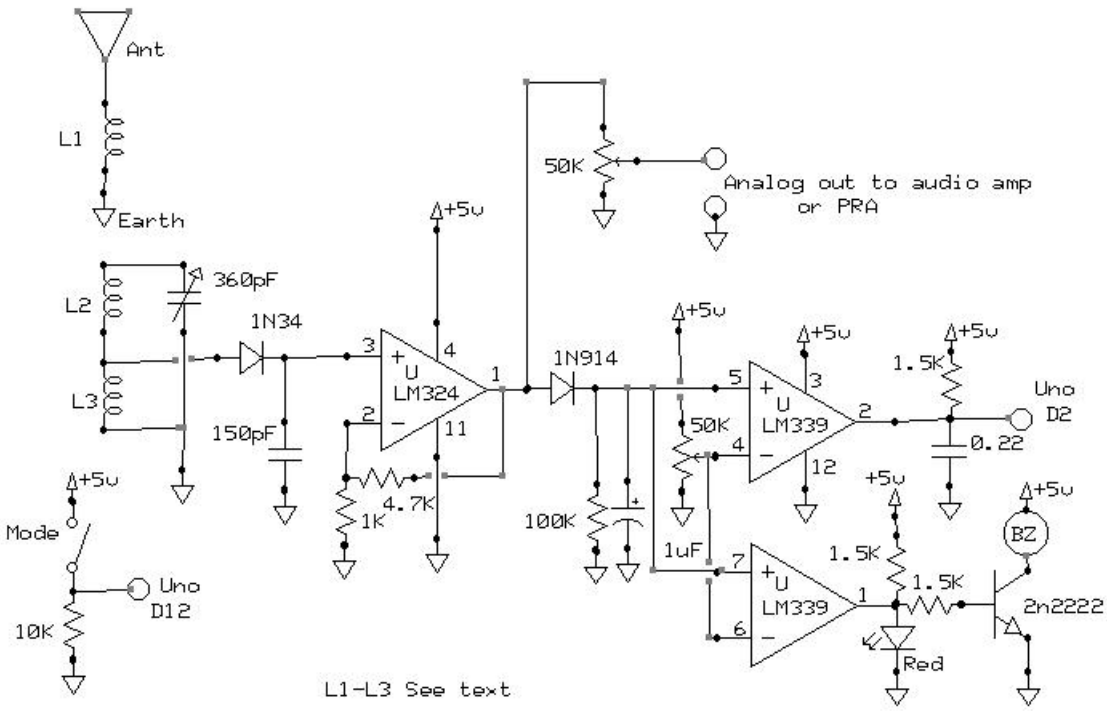


Antenna coil and 180uH tuning coil

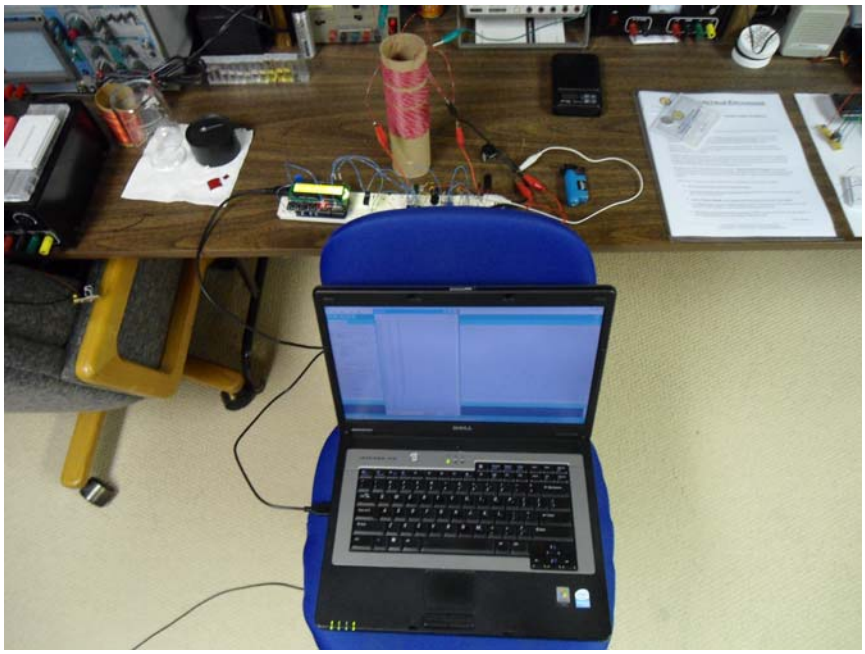
In the tank circuit with the 360pF variable capacitor (see schematic), the resonant frequency at the midpoint of the capacitor (150pF) is very nearly 1MHz, the middle of the AM Broadcast

Band. For lightning reception, I like tuning around the low end around 500KHz; seems less noisy there but you may wish to experiment. Select “quiet” spots away from broadcast stations. The 2 far ends of the tuning coil attach to the 360pF variable capacitor, with the end of the shorter 25 turn section circuit grounded. The tap connects to the 1N34 Germanium diode for demodulation. The 150pF capacitor smoothes out most of the carrier wave leaving the lower freq broadcast audio and lightning waves. Next, the LM324 single supply amplifier adds a gain of X4. It is powered by the Arduino’s +5/GND connections so no additional external power supply is needed. Outputs from the amp are to the 50K pot, then out to an optional audio amp, and out to the 1N914 rectifying diode and the filter to shape the output into a fast rising but slow decaying spike to filter out multiple lightning strokes, and concentrate on the single lightning flash.

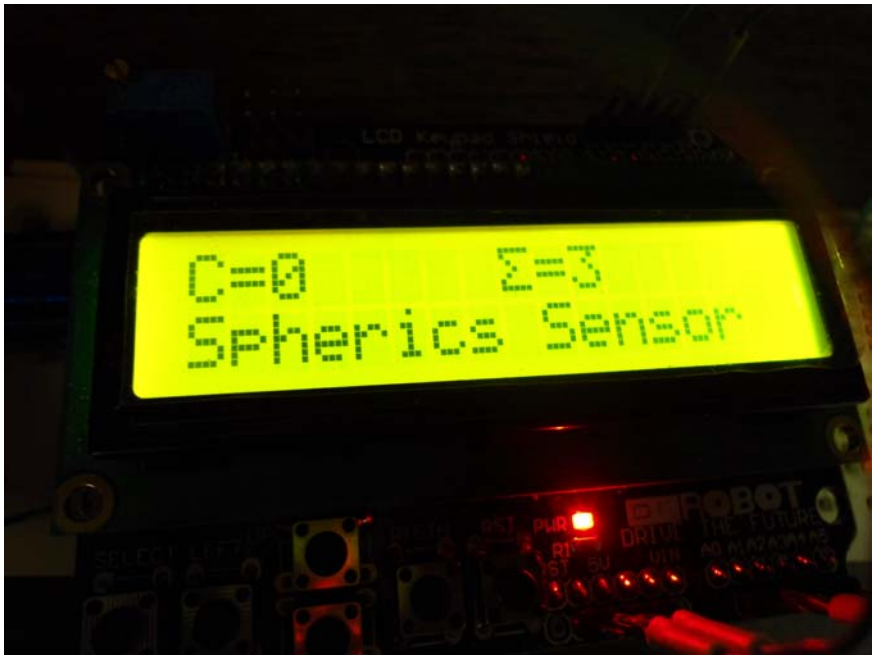
The filtered pulse is then applied to 2 comparators in an LM339 quad comparator to convert to clean squarewaves. One output is for indicators like an LED or buzzer, while the other is applied to Arduino digital pin 2 of the Uno.



Circuit Schematic



Whole system view with Crystal Radio, Arduino with LCD Keypad shield and Laptop for serial data.



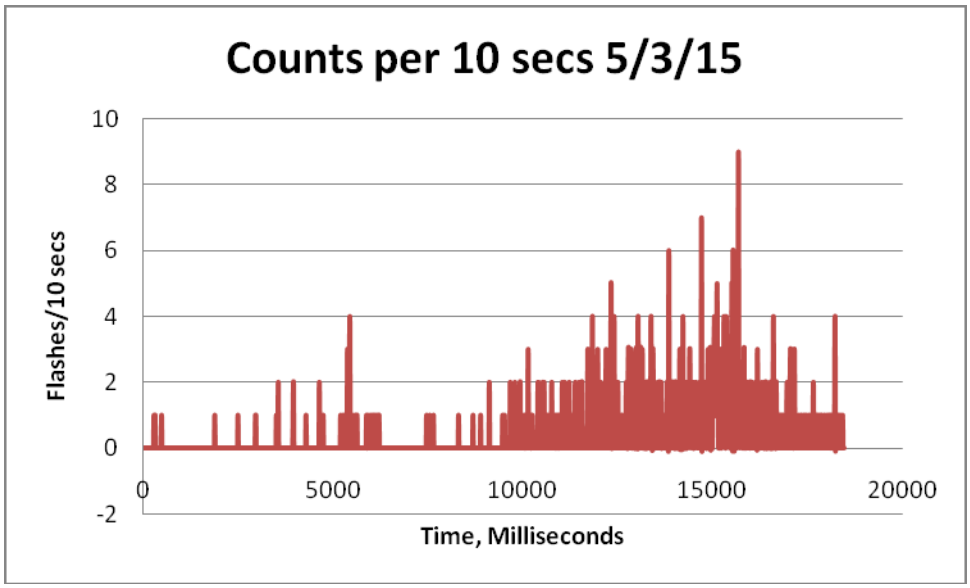
LCD Keypad data display of Counts per unit time and Total counts.

After the circuit is assembled and connected to the Uno, and the sketch loaded (see below), you're ready to capture the signature of a thunderstorm. The mode switch selects between flash event counting or no counting for AM radio listening. I tried 2 different counting time settings, 10 secs and 1 minute, and I like the 1 minute setting better, but feel free to experiment with the delay time and filter capacitor values. At one minute intervals, data can be collected for many hours without the amount of data getting too large. When ready, open the serial monitor. After a thunderstorm has passed, copy the data from the serial window. Open Excel and paste the data in. Use Data, Text to Columns, Comma separated to transfer it into usable columns. Then chart the results. An option on the audio output is to connect that output to a laptop running the freeware program PRA. It was intended for use with DIY gamma ray spectrometers, but can make histograms of pulses per time and amplitude from this pulse output as well. Connecting to

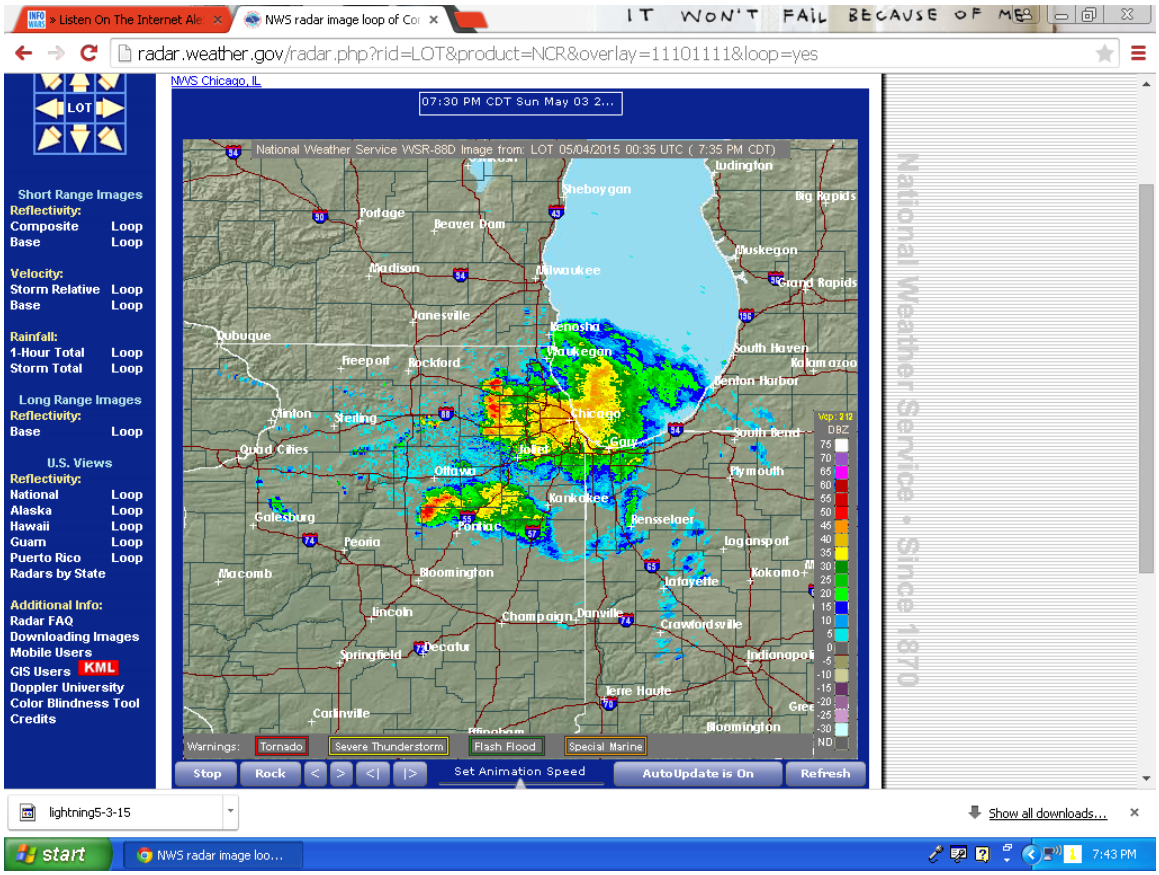
any small audio amplifier, including DIY with an LM386 amp allows listening to AM broadcast stations.

```
//LIGHTNING MONITOR NJC 5/9/15  Lightning pulses to D2, led output on D3
 //Mode switch on D12 V.8

#include <LiquidCrystal.h> // includes necessary library
int INPIN = 2;      // input pulses on pin D2

LiquidCrystal lcd(8, 9, 4, 5, 6, 7);  // connections to LCD
volatile long int numint = 0;  // counter for input pulses
int countotal=0;
int ledPin = 11;
int modePin = 12;                    // the number of the mode switch
int reading;



void intservice(void)  // this is called for every input rising edge
{
  numint++;
  digitalWrite(ledPin, HIGH);
  delay(200);
  digitalWrite(ledPin, LOW);
}



void setup() {
  pinMode(modePin, INPUT);           //pin 12 switch input

  pinMode(INPIN, INPUT);
  pinMode (ledPin, OUTPUT);
  digitalWrite(INPIN, LOW);
  attachInterrupt(0, intservice, RISING);  // attach interrupt routine
  lcd.begin(8,2);
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("  Lightning RF");
  lcd.setCursor(0, 1);
  lcd.print("Detector V.8 NJC");
  Serial.begin(9600);
  Serial.println("Arduino Lightning Detector V.8");
  Serial.println("ET(min), Flashes/min, Sigma");
  delay(5000);
}



void loop()
  {
  reading = digitalRead(modePin);           //check switch
  if (reading == HIGH)               //switch condition
  {
```

```
  countotal=countotal+numint;



  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("C=");
  lcd.print(numint);
  lcd.setCursor(8,0);
  lcd.print((char)246);              // sigma symbol
  lcd.print("=");
  lcd.print(countotal);
  lcd.setCursor(0,1);
  lcd.print("Spherics Sensor");
  Serial.print(millis()/60000);          // print elapsed minutes  to serial
  Serial.print(",    ");
  Serial.print(numint);            //print count/min to serial
  Serial.print(",    ");
  Serial.println(countotal);         //print total count to serial
  numint=0;                           // zero counter
  delay(60000);
}


else
{                                  //standby
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("  Lightning RF");
  lcd.setCursor(0,1);
  lcd.print("  AM Band RCVR");
  countotal=0;
  numint=0;
  delay(1000);
}
  }
```



First test, May 3 2015, logging counts every 10 seconds. Note intensity ramp-up.



NOAA radar on May 3. Note the area of lower intensity followed by higher intensity, matching the ramp-up seen on the Arduino RF monitor, as the system traveled from west to east..

(1) Uman, Martin A. , (1984) *Lightning.* New York, Dover Publications